A METHOD, SYSTEM, AND COMPUTER PROGRAM PRODUCT FOR GENERATING AND VERIFYING ISOLATION LOGIC MODULES IN DESIGN OF INTEGRATED CIRCUITS

DESCRIPTION

[Para 1] BACKGROUND OF THE INVENTION

[Para 2] Field of the Invention

[Para 3] The present invention relates generally to the design of integrated circuits (ICs) and more particularly to methods for generating and verifying isolation logic in the design of ICs.

[Para 4] Description of the Related Art

[Para 5] Complex electronic systems are typically designed using integrated circuits (ICs) comprising multiple functional blocks. An IC may have a wide range of power supply conditions, a number of independent power domains, and circuit performance objectives. Generally, different power domains are established between digital, analog and radio frequency (RF) functional blocks on an IC. As an example, a wireless handset chip has several power domains due to multiple modes of operation.

[Para 6] Referring to Fig. 1, an exemplary circuit 100 including two power domains 110 and 120 is shown. The power domains 110 and 120 are logic units serving different functions. Power domains 110 and 120 are powered by signals VC_1 and VC_2 supplied by a level shifter 130, which translates power signals between two voltage domains. For example, level shifter 130 may translate signals originating from a first domain operating under a lower supply voltage (e.g., 1.2V) to a second domain operating with a higher supply voltage (e.g., 2.5V). However, it should be noted that power domains do not have to be connected to a level shifter.

[Para 7] For power management purposes and reducing power consumption, parts of a design are usually turned off during the operation of a semiconductor device. Specifically, power domains that power IC areas not

actively used in certain modes of operation are completely shut down. A correct design requires that when a power domain is shut down, its output signals will not become indeterminable so that an unknown state is transferred to the rest of the design.

[Para 8] Moreover, leakage power is a critical concern for design of ICs that operate in stand-by mode and are manufactured using advanced fabrication technologies, such as 90nm and below. Turning off a supply to the regions of design in such modes of operation eliminates leakage power consumption associated with these regions completely. For isolating and enforcing stable output values at shutdown, the outputs of power domains 110 and 120 are connected to isolation logic modules 140 and 150 respectively. Isolation logic modules 140 and 150 ensure that power domains 110 and 120 are correctly isolated and none of their outputs is left indeterminable, and are therefore determinable, when the power is off. The design of isolation logic demands a designer to determine a set of shutdown constraints including steady state output values, shutdown conditions, wakeup/shutdown signals, and so on.

[Para 9] Prior art design tools, e.g., computer aided design (CAD) do not provide automated means for isolating power domain in the design, i.e., generating and inserting isolation logic modules in the design. Moreover, such tools generally require that the user identify all power domains in the design, define an isolation logic for each domain and check its correctness. In ICs where the number of power domains may be large, this is an inefficient, time-consuming, as well as an error prone task.

[Para 10] It would be, therefore, advantageous to provide a solution that automatically generates isolation logic modules for power domains and appropriately places these modules in the design. It would be further advantageous if the provided solution automatically detects isolation logic modules within the design and checks their correctness.

[Para 11] SUMMARY OF THE INVENTION

[Para 12] The present invention addresses the aforementioned problems. An aspect of the invention is to provide a method for generating and verifying isolation logic modules in a design of an integrated circuit (IC), the method

comprising specifying a plurality of voltage constraints defining at least one power domain in the design, iteratively checking, for each of the power domain, if an isolation logic module isolating the power domain exists in the design, verifying the correctness of the isolation logic module existing in the design, if the isolation logic module is identified, generating an isolation logic module for isolating the power domain, if the power domain is not correctly isolated, and inserting the generated isolation logic in the design.

[Para 13] Consistent with an aspect of the present invention, there is provided a computer program product, comprising a computer-readable medium with instructions to enable a computer to implement a method for generating and verifying isolation logic modules in a design of an integrated circuit (IC), the method comprising specifying a plurality of voltage constraints defining at least one power domain in the design, iteratively checking for each of the power domain if an isolation logic module isolating the power domain exists in the design, verifying the correctness of the isolation logic module existing in the design, if the isolation logic module is identified, generating an isolation logic module for isolating the power domain, if the power domain is not correctly isolated, and inserting the generated isolation logic in the design.

[Para 14] Consistent with another aspect of the invention, there is provided a method for generating isolation logic modules in a design of an integrated circuit (IC), the method comprising specifying a plurality of voltage constraints defining at least one power domain in the design, iteratively producing, for each of the power domain using the voltage constraints, a description language code implementing the isolation logic module, instantiating the description language code to form an instance of the isolation logic module, inserting the instance of the isolation logic module in a wakeup domain, renaming output names of the power domain, and assigning the original names of the power domain's output names to outputs of the isolation logic module.

[Para 15] In another aspect, there is provided a computer program product, including a computer-readable medium with instructions to enable a computer to implement a method for generating isolation logic modules in a design of

an integrated circuit (IC), the method comprising specifying a plurality of voltage constraints defining at least one power domain in the design, iteratively producing, for each of the power domain using the voltage constraints, a description language code implementing the isolation logic module, instantiating the description language code to form an instance of the isolation logic module, inserting the instance of the isolation logic module in a wakeup domain, renaming output names of the power domain, and assigning the original names of the power domain's output names to outputs of the isolation logic module.

[Para 16] Consistent with an aspect of the present invention, there is provided a method for verifying the correctness of isolation logic modules in a design of an integrated circuit (IC), the method comprising specifying a plurality of voltage constraints defining at least one power domain in the design, iteratively simulating shutdown conditions for each of the power domain, comparing each of the output values of the power domain to a respective steady state value, and generating an error report if the comparison results in an equality, checking if at least one isolation cell in the isolation module is not connected to a wakeup/shutdown signal, and generating the error report if the checking results in an affirmative answer, and checking if the wakeup/shutdown signal is generated in a wakeup domain, and generating the error report if the checking results in a negative answer; otherwise, generating a success report.

[Para 17] In another aspect of the invention, there is provided a computer program product, including a computer-readable medium with instructions to enable a computer to implement a method for verifying the correctness of isolation logic modules in a design of an integrated circuit (IC), and the method comprises specifying a plurality of voltage constraints defining at least one power domain in the design, iteratively simulating shutdown conditions, for each of the power domain, comparing each of the output values of the power domain to a respective steady state value, and generating an error report if the comparison results in an inequality, checking if at least one isolation cell in the isolation module is not connected to a wakeup/shutdown

signal, and generating the error report if the checking results an affirmative answer, and checking if the wakeup/shutdown signal is generated in a wakeup domain, and generating the error report if the checking results in a negative answer; otherwise, generating a success report.

[Para 18] In another aspect, there is provided a computer system for generating and verifying isolation logic modules in the design of integrated circuit (IC), the system comprising a processor and a memory under control of the processor, a database operable to maintain voltage constraints specified by a user, a code generator operable to generate description language code of the isolation logic modules, an insertion unit operable to instantiate and insert in each of the isolation modules a respective wakeup domain, a checking unit operable to verify the correctness of the isolation logic modules, and a simulator for simulating shutdown conditions.

[Para 19] BRIEF DESCRIPTION OF THE DRAWINGS

[Para 20] Figure 1 is an exemplary circuit including two power domains (prior art);

[Para 21] Figure 2 is a non-limiting flowchart describing the operation of the present invention;

[Para 22] Figure 3 is a non-limiting flowchart describing the procedure for generating and inserting isolation logic consistent with an exemplary embodiment of the present invention;

[Para 23] Figures 4A-4D are non-limiting examples for generating and inserting isolation logic;

[Para 24] Figure 5 is a non-limiting flowchart describing the procedure for checking isolation logic consistent with an exemplary embodiment of the present invention; and

[Para 25] Figure 6 is an exemplary implementation of a system for verifying power domains in the design of an integrated circuit (IC).

[Para 26] DETAILED DESCRIPTION OF THE INVENTION

[Para 27] An exemplary embodiment of the present invention generates an isolation logic module for each power domain specified by a user, instantiates the created module in a specified wakeup domain, and then simulates the shutdown conditions to ensure the correctness of a generated isolation logic module against the specified values. The isolation logic is generated based on user-defined voltage constraints.

[Para 28] Referring to Fig. 2 a non-limiting flowchart 200 describing a method disclosed by the exemplary embodiment of the present invention is shown. For the purpose of verifying if power domains are correctly isolated, various constraints (hereinafter the "voltage constraints") are specified by a user (e.g., a design engineer) using, for example, a graphical user interface (GUI) at step S210. The voltage constraints include, but are not limited to, at least one power domain to be tested, its corresponding wakeup domain, a wakeup/shutdown signal and a list of steady state values. The wakeup/shutdown signal is generated in a wakeup domain and is enabled when the power domain is off. For example, the voltage constraints can be defined as follows:

[Para 29] (1). voltagedomain -instname "top.wkup_domain_inst+" -name WPD -values 1.0

[Para 30] (2). voltagedomain -instname "top.megamodule1+" -name SBPD1 -values 1.0 0.0 --isosig pden1 -isoval 0

[Para 31] (3). voltagedomain -instname "top.megamodule2+" -name SBPD2 -values 1.2 0.0 -isosig pden2 -isoval 1

[Para 32] The constraint defined in (1) is a voltage-domain called "WPD", which is a wakeup domain of power domains SBPD1 and SBPD2 specified in constraints (2) and (3) above. The wakeup domain WPD is always on, as opposed to the domains SBPD1 and SBPD2 which are shut down during certain modes of operation. The 'isosig' argument determines the wakeup/shutdown signal and its corresponding values.

[Para 33] At step S215, a single power domain to be verified is selected. A different power domain is selected each time execution reaches this step to

ensure that all power domains specified in the voltage constraints are tested. At step S220, it is determined whether an isolation logic isolating the selected power domain exists in the design, and if so execution continues with S230 where the correctness of this isolation logic is tested; otherwise, execution continues with step S240 where a procedure for generating and inserting an isolation logic is applied. Specifically, this procedure generates a description language code of a module defining the isolation logic for each power domain specified by the user and instantiates this module in the specified wakeup domain. The description language may be, but is not limited to, Verilog, VHDL (VHSIC Hardware Description Language), or a combination thereof. The procedure for generating a power domain is described in greater detail below.

[Para 34] Once the isolation logic is created, execution continues with step S230 for checking the correctness of the created logic. For the purpose of checking the isolation logic, shutdown conditions are simulated and the outputs of the power domains connected to the under-test isolation logic are examined. The isolation logic is considered correct if each of those outputs produce a well-determined value, i.e., either '1' or '0'. Likewise, the value prior to shut down can be retained through a retention cell. Step S230 is executed by a checking procedure described in further detail below.

[Para 35] At step S250, it is determined if the check succeeded, and if so at step S270 a success report including the check results is sent to the user; otherwise, execution continues with step S260 where an attempt is made to correct the isolation logic. Specifically, the register transfer level (RTL) statements are analyzed to resolve the error or errors in the design detected by the checking procedure. For example, if the check discovers that one of the isolation cells forming the isolation logic is not connected to a common enable signal, then the design is fixed by connecting that isolation cell to the common enable signal. The execution then continues with step S230. At step S280, another check is made to determine if all power domains specified in the voltage constraints were handled and if so, execution ends; otherwise, execution continues with step S215.

[Para 36] Referring to Fig. 3, a non-limiting flowchart S240 describing the procedure for generating and inserting isolation logic consistent with an exemplary embodiment of the present invention is shown. At step S310, the user specifies a steady state value for each output of the currently handled power domain. The output values are defined as pairs of <name, value>, i.e., the hierarchical name of an output signal is specified along with its steady state value. The steady state value may be either '0' or '1'. For a bus, either the value of each bit or the value of all bits can be specified. For example, a bus such as a DATA bus, where most of the bits hold the same steady state value, while some sparse bits differ, the user can choose to override the values of the sparse bits.

[Para 37] At step S320, the user may define the isolation cells that form the isolation logic. An isolation cell may be, but is not limited to an AND gate, an OR gate, a Latch, and the like. For isolation cells the user may define the enable signal for that cell and its value (i.e., "active high" or "active low"). In an exemplary embodiment, the steady state values and the isolation logic are part of the voltage constraints.

[Para 38] At step S330, based on the voltage constraints defined by the user, the isolation logic is generated. Specifically, a description language code module implementing the isolation logic for a selected power domain is produced. This code comprises instructions assuring that the power domain is correctly isolated. That is, under power off conditions the output values of the power domain are as the pre-determined steady state values. The code module further comprises instructions for determining the wakeup/shutdown signal that enables or disenables the isolation logic. An exemplary Verilog code module defining isolation logic is provided below.

[Para 39] At step S340, the module generated at step S330 is instantiated and inserted into a wakeup domain specified by the user. The method correctly instantiates the isolation logic modules in the wakeup domains using back referencing analysis and a synthesized netlist, which generally includes logical gates such as AND, NAND, NOR, OR, XOR, XNOR, latches, and the like. The back referencing analysis provides the precise location of the power domain

instances in the design. The back referencing analysis connects a synthesized netlist object model with a data model of the netlist. For every definition and use of a signal in the netlist object model, a means of cross probing is established and the line and file name of every such definition is stored. The same operation is also performed for each definition and instantiation module. Thus, given a definition of use of a signal in the netlist object model, the precise location of this signal in the design file can immediately be established. Similarly, given an instantiation of a module in the synthesized object model, a corresponding location in the data model can be retrieved.

[Para 40] To restrict the locality of changes, the outputs names of the power domains are renamed and the original names are retained as outputs names of the isolation module. This ensures compatibility with signals feeding into other wakeup or "always on" domains. The synthesized netlist (or gate level netlist) may be produced by an IC synthesis tool. Synthesis tools produce a gate level netlist based on the RTL statements. One such synthesis tool is disclosed in a US patent application entitled "An Apparatus and Method for Handling of Multi-Level Circuit Design Data", serial number 10/118,242, having a common assignee with the present invention and is hereby incorporated by reference.

[Para 41] At step S350 new design files including the new isolation logic are generated and displayed to the user. The design files may include a new RTL description and the synthesized netlist.

[Para 42] Figs. 4A-4D show a non-limiting example for generating and inserting an isolation logic. In this example, isolation logic for a power domain named "SPWD1" <MBS: SPWD1 is just a name given to the module> that includes a memory controller "mc_top" 410 is generated. The specified wakeup/shutdown signal is "susp_req_i" and is enabled at a logic level '1'. The memory controller, as shown in Fig. 4A, includes the following outputs: wb_ack_o, wb_err_o, wb_read_go, wb_write_go, wb_first, wb_wait, wr_hold, and wr_data_o. The steady state values, defined using the notation <name, value>, are as follows: <mc_top.wb_ack_o, 1>, <mc_top.wb_err_o, 0>, <mc_top.wb_read_go, 0>, <mc_top.wb_write_go, 0>, <mc_top.wb_first, 1>, <mc_top.wb_wait, 1>, <mc_top.wr_hold, 0>, and <mc_top.wb_data_o, 0>.

The wr_data_o output is a data bus and the steady state value '0' is the same for all bits in the bus.

[Para 43] Once the steady state values are determined, the VHDL or Verilog code module for the isolation logic is created. An exemplary Verilog code of isolation logic generated module SPWD1 is provided in Fig. 4B. The input and output statements (shown in lines 4010 through 4170 of the exemplary code) represent connections to the isolation module. The input "iso_signal_blocking" is the port name for wakeup/shutdown signal. When the value at this input equals '1' the isolation module is enabled.

[Para 44] The assignment statements (shown in lines 4190 through 4580) are programmed to ensure that the steady state values at shutdown are as defined by the user. An "assign" statement denotes a concurrent continuous assignment, which describes the functionality of the module. A concurrent assignment executes whenever one of the inputs changes value. For instance, the steady state value defined for the output "wb_wait" is '1'. The isolation logic sets (as shown in line 4210) the value of this output (expressed as "out_wb_wait" in the isolation module) to be equal to the outcome of the logical function: iso_signal_blocking OR wb_wait. The value of the iso_siganl_blocking at shutdown is always '1' and thus, at shutdown, the value of "out_wb_wait" is always equal to '1'. This can be easily seen in the timing diagram depicted in Fig. 4D.

[Para 45] Thereafter, the Verilog isolation module is instantiated in the specified wakeup of the power domain. An instance of the generated isolation logic module "iso_logic for SPWD1" 420 is provided in Fig. 4C. As shown in Fig 4C, the isolation logic instance 420 includes all the outputs and inputs defined in the Verilog code shown in Fig. 4B.

[Para 46] Referring to Fig. 5 a non-limiting flowchart S230 describing the checking procedure consistent with an exemplary embodiment of this invention is shown. This method can be used to verify the correctness of isolation logic rendered by the generating procedure or isolation logic that already exists in the design.

[Para 47] At step S510, the shutdown conditions are simulated, i.e., the power to the power domain connected to the isolation logic under test is shut off and the wakeup/shutdown signal is enabled.

[Para 48] At step S520, a check is made to determine if the outputs of the power domain are correctly isolated. Specifically, the value of each output under the shutdown conditions is compared with a respective steady state value specified by the user. When such values are not specified, it is checked that each output is not floated. If the check results in an error such as the quality of the compared values, at step S550 a report including the error type or the cause of the error is generated and sent to the user; otherwise, execution continues with step S530, where another check is made to determine if each isolation cell forming the isolation logic is connected to a common enabling signal, i.e., the wakeup/shutdown signal. For example, it is checked whether the "iso_signal_blocking" is connected to each isolation cell forming the isolation logic "iso_logic_for SPWD1". If step S530 results in an error, at step S550 an error report including the error type is generated and sent to the user; otherwise, execution continues with step S540 where another check is performed in order to verify that the wakeup/shutdown is generated in a wakeup domain, and if so at step S560 a success report including the test result is generated and sent to the user; otherwise, execution continues with step S550. It should be noted that if one of the outputs of a power domain is connected to a different power domain, it is checked whether a level shifter is connected between the crossing identified power domains.

[Para 49] Consistent with an exemplary embodiment of the present invention the simulated shutdown conditions and the outputs tested during the execution of the checking procedure are highlighted in the design by means of a visualization tool. For example, a '0' value may be highlighted in orange, a '1' value in blue, and a floating output is highlighted in yellow. If a node is left floating, then the associated error code in VHDL or Verliog is also highlighted along with the highlighting of an associated schematic. This provides the user with an easy way to immediately recognize the floating outputs in the design, and thus save debugging time.

[Para 50] The methods disclosed can be further embodied by a person skilled in the art as part of a computer software program, a computer aided design (CAD) system, a CAD program, a netlist voltage domain analysis tool, and a RTL voltage domain analysis tool, and the like.

[Para 51] Referring to Fig. 6 an exemplary implementation of a system 600 for generating and verifying isolation logic modules in the design of integrated circuits (ICs) is shown. A database 610 maintains the voltage constraints specified by the user. A code generator 620 generates a description language code (e.g., VHDL, Verilog, or combination thereof) of isolation logic modules. An insertion unit 630 instantiates and inserts the isolation modules in the specified wakeup domains. The insertion unit 630 also outputs the updated design files. A checking unit 640 checks the correctness of the generated isolation logic and sends reports, including the check results, to the user. The isolation logic to be tested is received from insertion unit 630 while the shutdown conditions are simulated by a simulator 650.

[Para 52] The invention has now been explained with reference to exemplary embodiments. Other embodiments will be apparent to those of ordinary skill in the art in light of this disclosure. The scope of the invention should not be thought of as being limited by the exemplary embodiments; rather, the appended claims should be consulted.